UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

Herramienta para monitorización y detección de cambios de modo automático sobre la superficie de la Tierra con imágenes satélite multiespectrales

Automatic monitoring and change detection tool using multispectral satellite images on Earth surface

Realizado por
Antonio Manuel Burgueño Romero

Tutorizado por
José Manuel García Nieto,
Antonio Jesús Nebro Urbaneja

Departamento
Lenguajes y Ciencias de la Computación
Universidad de Málaga

MÁLAGA, FEBRERO, 2020

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

Herramienta para monitorización y detección de cambios de modo automático sobre la superficie de la Tierra con imágenes satélite multiespectrales

Automatic monitoring and change detection tool using multispectral satellite images on Earth surface

Realizado por:

Antonio Manuel Burgueño Romero

Tutorizado por:

José Manuel García Nieto, Antonio Jesús Nebro Urbaneja

Departamento:

Lenguajes y Ciencias de la Computación

MÁLAGA, FEBRERO, 2020

Fecha defensa:

Fdo.: El Secretario del Tribunal

Resumen

El principal objetivo de este TFG es la realización de una herramienta que clasifique automáticamente áreas de interés en el marco de imágenes de satélites para observación de la Tierra. Al clasificar estas imágenes, es posible monitorizar cambios en el terreno, detectar incendios, etc. Con la ayuda de las imágenes multiespectrales, realizadas por la constelación de satélites Sentinel-2 (siendo estas imágenes de libre acceso) podemos realizar estudios sobre casi cualquier localización. Con el objetivo de automatizar el proceso, se desarrolla una aplicación en el lenguaje de programación Python, que abstrae la paralelización del proceso y los problemas que surgen al tratar con imágenes de gran resolución respecto al usuario final. Para poner a prueba la herramienta, se realizan dos estudios en los que se clasifican zonas con distinta composición de clases: una rural y otra metropolitana. Posteriormente se realizan análisis predictivos sobre esas zonas.

Palabras claves: Cobertura terrestre, imágenes multiespectrales, clasificación del terreno, Sentinel-2, Python.

Abstract

The main goal of this Undergraduate Thesis is to develop a tool that will be able to automatically classify regions of interest in the context of Earth observation using satellite imagery. Currently only a few tools are avaiable to treat this kind of data, which may be important for fields as geology or biology. It is possible to perform land cover studies with the help of open-data multispectral images taken by Satellite constelation Sentinel-2. A Python application is developed in order to achieve automatization. It deals with parallelization and data size problems, making it easy to use for the final user.

Keywords: Land cover, multispectral imaging, supervised learning, remote sensing, Sentinel-2, Python.

Índice general

Capítulo 1

Introducción

La necesidad de la aparición de nuevas herramientas que apoyen temas como la agricultura de precisión o la lucha contra el cambio climático es cada vez más urgente. En este trabajo de fin de grado se va a tratar de dar uso a tecnologías avanzadas como el análisis de datos en imágenes satélite multiespectrales, buscando así realizar avances en ambos campos.

A continuación se va a a proporcionar una introducción a la materia en la que se basa éste proyecto, los motivos que me hicieron ver necesario el desarrollo del mismo y los objetivos que se pretenden cumplir al finalizarlo.

1.1. Contexto y motivación

Cada año hay más satélites dedicados a la observación de la tierra, cada uno de los cuales genera grandes cantidades de datos de acceso libre al día. La necesidad de herramientas que traten estos datos y generen información a partir de ellos es cada vez mayor.

La constelación Sentinel-2 tiene dos satélites pertenecientes al programa Copérnico de la ESA[1] (European Space Agency) que llevan desde 2015 capturando imáge-

[1] `https://www.esa.int/Applications/Observing_the_Earth/Copernicus`

nes multiespectrales de la superficie terrestre. Las imágenes multiespectrales, a diferencia de las imágenes RGB, aportan información de otras bandas fuera del umbral visible.

En este proyecto se propone una herramienta para procesar y analizar los datos de los satélites Sentinel-2. Cabe destacar que la herramienta puede procesar cualquier imagen multiespectral; no importa que sea procedente de Sentinel-2, de cualquier otro satélite de observación terrestre o de imágenes realizadas por drones.

Al ser estas imágenes de libre uso para fines científicos, se abre una amplia posibilidad para realizar estudios y análisis sobre los datos. Todos estos recursos se pueden usar para teledetección de cambios en cultivos, monitorización de regeneración de zonas verdes tras incendios forestales, clasificación de la superficie terrestre, detección de anomalías, etc.

Sentinel-2 tiene un ciclo de repetición de cinco días, por lo que en una localización dada se dispone de una nueva imagen en ese período. Los datos de Sentinel-2 se organizan en productos de 10 000 km^2 en el que cada píxel tiene una resolución espacial de 10, 20 o 60 metros, dependiendo del rango espectral. Estos productos incluyen información para 13 bandas que van desde los 443nm a los 2190nm. Cada producto tiene un tamaño medio de 700Mb.

Queda claro, por tanto, que existe una demanda de herramientas que puedan manejar grandes cantidades de datos de forma eficiente en el ámbito de tratamiento de imágenes satélite, ya que cada día se obtienen cientos de terabytes de acceso libre. El mismo Sentinel Hub, servicio Web desarrollado por la ESA, contiene varias herramientas de este estilo, como puede verse en la Figura 1.1. Esto nos permite calcular índices espectrales para la monitorización de áreas vegetales, humedad, propiedades del terreno, etc, lo que nos proporciona capacidad para realizar análisis predictivo.

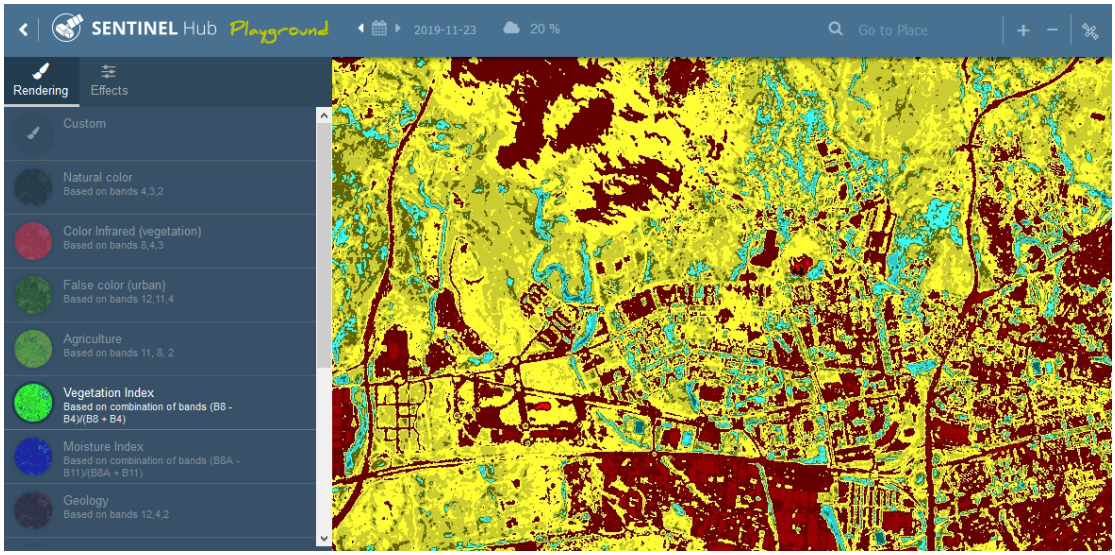


Figura 1.1: Servicio web para la observación terrestre usando productos Sentinel-2, ofrecido por la ESA. Permite al usuario tomar un primer contacto con las imágenes multiespectrales que captan sus satélites.

Fuente: `https://apps.sentinel-hub.com/sentinel-playground/`

1.2. Objetivos

Objetivo principal

El principal objetivo de este proyecto es facilitar una herramienta para el tratamiento y análisis de imágenes satélite, que sería útil para monitorización de superficies terrestres, detección de cambios en el terreno, agricultura de precisión, etc.

Objetivos específicos

- **Clasificación**: Clasificar automáticamente cualquier área de interés en la corteza terrestre. Es decir, a cada píxel de las imágenes multiespectrales que tenemos de la zona deseada, se le asigna automáticamente una clase previamente definida (aprendizaje supervisado).

- **Máscaras de nube**: Uno de los grandes problemas que dificultan la clasificación es la presencia de nubes en las imágenes, que corrompen la información de los píxeles. La detección automática de nubes aún no ha sido lograda con precisión, ya que tienen unas características muy heterogéneas. Por tanto, otro de los objetivos es obtener una máscara de nubes, para que el clasificador tenga en cuanta cuándo puede aprender con un píxel y cuándo no.

- **Casos de uso**: Una vez obtenida la imagen clasificada, es mucho más fácil la realización de estudios sobre el área. Otro de los objetivos es el desarrollo de varios casos de uso que pongan a prueba el funcionamiento de la herramienta. Se calculará el área de zona verde en Teatinos y de agua en el embalse de Iznájar en los últimos años, pudiéndose así analizar sus cambios en el tiempo y dar lugar a una posible predicción de éstos en el futuro.

1.3. Estructura del documento

El presente documento se encuentra divido en varios capítulos, recogiendo toda la información relacionada con el desarrollo de este proyecto. En el capítulo 2 se pone en contexto el estado del arte actual en temas de clasificación de imágenes de satélite, además se explican varios conceptos básicos de la materia.

En los capítulo 3 y 4 se comentan la metodología de desarrollo de software usada y se va explicando el diseño de la herramienta. En el capítulo 5 se analizan los resultados obtenidos para los dos casos de uso que se realizan; además se habla de los problemas con los que hay que lidiar en el desarrollo de una aplicación que trate con teledetección e imágenes satélite multiespectrales.

Finalmente, en el capítulo 6 se comenta el futuro de este tipo de aplicaciones y de la observación terrestre en general. También se habla del futuro de esta herramienta y cómo podría mejorar.

Capítulo 2

Conceptos previos y estado del arte

En este proyecto se usan imágenes multiespectrales, captadas por los satélites Sentinel-2 mediante técnicas de teledetección. Se van a poner en contexto las tecnologías usadas en la captación de las imágenes, así como las demás usadas reiterativamente en el proyecto. Además, se pondrá en contexto el estado del arte sobre la clasificación de imágenes satélite.

2.1. Conceptos

2.1.1. Imágenes captadas por Sentinel-2

A continuación se van a definir aspectos que necesitamos entender cuando tratamos con imágenes Sentinel-2; la mayoría son parámetros de los sensores del satélite. Estos satélites generan imágenes multiespectrales, que son aquellas en las que se captan datos para específicos intervalos del espectro electromagnético. Así como las imágenes RGB tienen tres bandas, las imágenes de Sentinel-2 se componen de 13 bandas espectrales, descritas en las figuras 2.1 y 2.2. Aunque en la Figura 2.1 solo se observa la resolución espacial, en realidad existen 4 tipos de resoluciones en las imágenes satélite:

Banda	Resolución Espacial	Longitud de onda central	Descripción
B1	60 m	443 nm	Ultra azul (Costa y Aerosol)
B2	10 m	490 nm	Azul
B3	10 m	560 nm	Verde
B4	10 m	665 nm	Rojo
B5	20 m	705 nm	Visible e Infrarrojo Cercano (VNIR)
B6	20 m	740 nm	Visible e Infrarrojo Cercano (VNIR)
B7	20 m	783 nm	Visible e Infrarrojo Cercano (VNIR)
B8	10 m	842 nm	Visible e Infrarrojo Cercano (VNIR)
B8a	20 m	865 nm	Rojo de borde (RedEdge)
B9	60 m	940 nm	Vapor de Agua
B10	60 m	1375 nm	Cirrus
B11	20 m	1610 nm	Onda Corta Infrarroja (SWIR)
B12	20 m	2190 nm	Onda Corta Infrarroja (SWIR)

Figura 2.1: Descripción de las 13 bandas de los satélites Sentinel-2

Fuente: `https://naturgis.es/2018/04/23/`

Espacial Área que cubre un píxel de la imagen. Si una banda espectral tiene una resolución espacial de lm, entonces el área que cubre cada píxel es un cuadrado de lado l (lxlm^2).

Temporal Frecuencia por la que algún satélite en órbita de la constelación pasa una localización particular. En Sentinel-2 suele ser de 5 días.

Radiométrica El número de diferentes intensidades de radiación que el sensor es capaz de distinguir.

Espectral En el caso de las imágenes Sentinel-2 son los intervalos del espectro electromagnético cubiertos por cada banda espectral (ver figura 2.2)

En realidad lo que miden los sensores de los satélites es la radiancia espectral, que es la radiación térmica que emite un cuerpo para todo el espectro electromagnético. La radiación que emite el cuerpo no es la misma que la que llega a los sensores del satélite; esto ocurre por la presencia de la atmósfera (entre otros motivos).

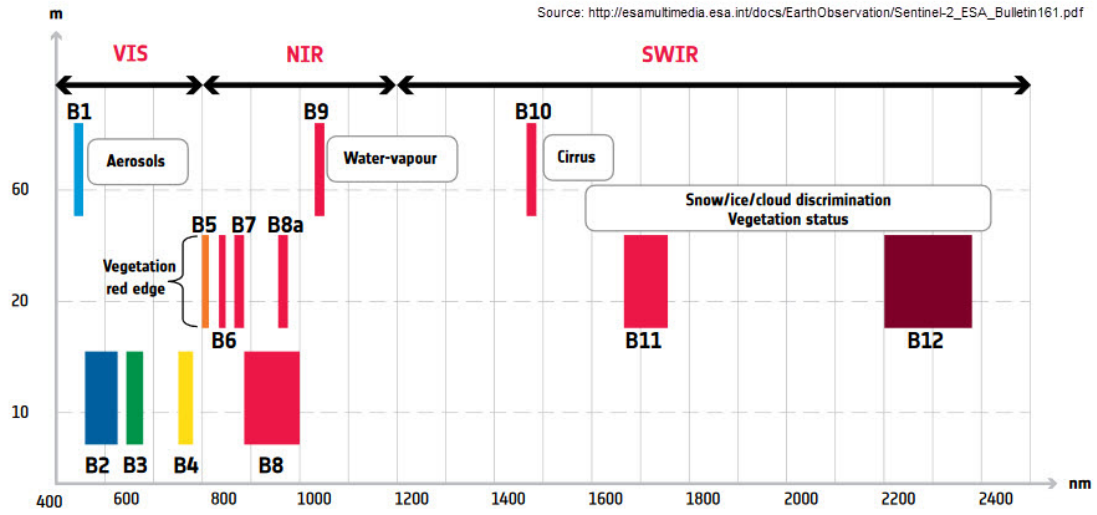


Figura 2.2: Resolución espectral y espacial de las bandas de las imágenes Sentinel-2

Al dato que toma el sensor se le llama **TOA (Top Of Atmosphere)**. En los productos Sentinel-2 se realiza un paso previo que es la transformación desde radiancia TOA a reflectancia TOA, que es una transformación directa y simple.

La reflectancia **BOA (bottom of atmosphere)** es la reflectancia que se obtendría si el satélite estuviera bajo la atmósfera (sin distorsiones). Es el dato más correcto y se puede predecir mediante ciertas técnicas más complejas.

La **firma espectral** de un objeto es simplemente su radiancia espectral. Se le llama firma ya que define las características del objeto. Para dos muestras de una plantación de trigo, sus radiancias espectrales van a ser muy similares. La firma espectral del trigo se podría estimar realizando una media entre una gran cantidad de muestras.

Como ya mencionamos antes, las imágenes Sentinel-2 se componen de 13 bandas espectrales. A cada píxel se le corresponden 13 valores asociados a un intervalo del espectro electromagnético. Por tanto, nuestras firmas espectrales serán un conjunto de puntos discreto en vez de una función continua (Figura 2.3).

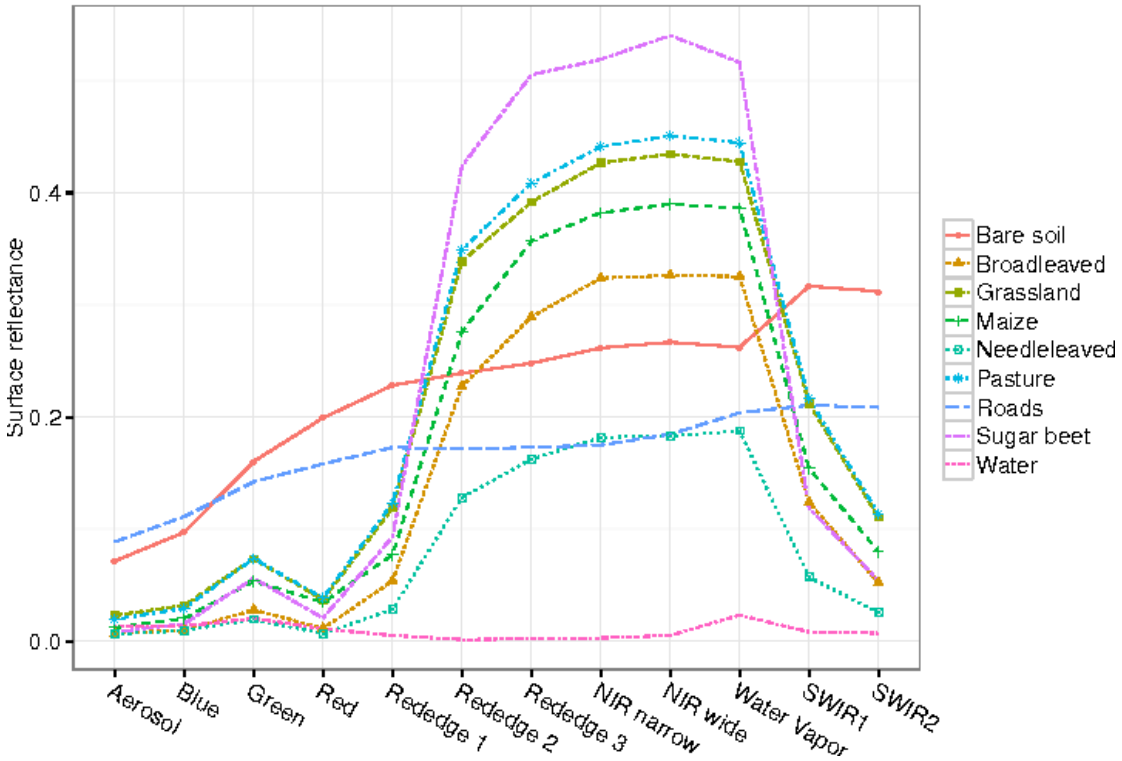


Figura 2.3: Firmas espectrales de varias muestras captadas por los sensores de los satélites Sentinel-2

Fuente: Radoux et al. [5]

Los **índices espectrales** son imágenes que se forman combinando bandas, normalmente se utilizan para distinguir ciertos objetos de otros. Uno de los índices más usados es el NDVI (índice de vegetación de diferencia normalizada), que se usa para distinguir la vegetación de otros terrenos:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

En Sentinel-2 NIR y RED son las bandas 8 (infrarrojo cercano) y 4 (rojo), respectivamente. Esto tiene sentido ya que si observamos la firma espectral de cualquier vegetación (Figura 2.4), se ve claramente el salto que hay entre la radiancia de ondas en el intervalo de frecuencia del rojo y el infrarrojo cercano. Por tanto, el NDVI será muy grande en estos casos (cercano a 1) y pequeño en otros objetos.

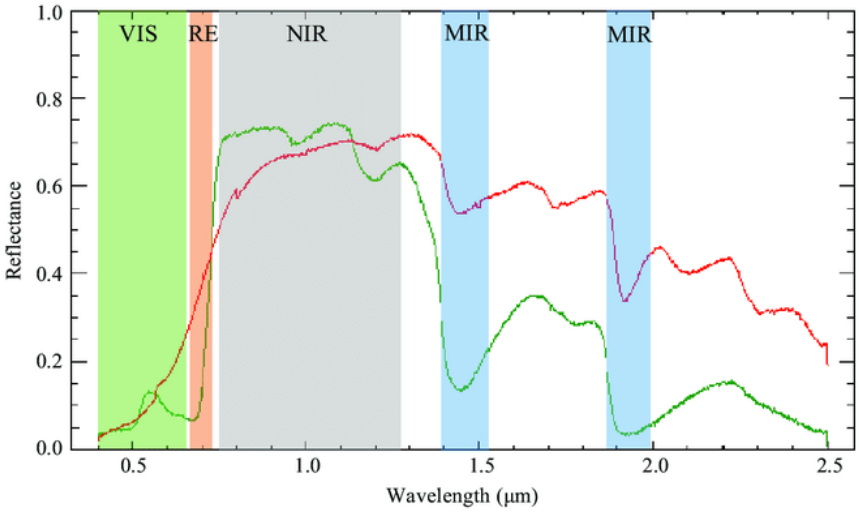


Figura 2.4: Firmas espectrales de muestras de hierba verde (en verde) y de hierba seca (en rojo)

Hay tantos índices espectrales como combinaciones de bandas, cada uno tiene un propósito propio y cada día expertos proponen nuevos y mejorados índices espectrales. Podemos ver una amplia lista de índices para Sentinel-2 en **indexdatabase** [1]

2.1.2. Estructura de datos: productos Sentinel-2

Descritas las imágenes y las bandas que las forman, vamos a ver a continuación la estructura de los datos en la que están contenidas. Cuando descargamos las imágenes, estas vienen envueltas en los llamados **productos Sentinel-2**. Los productos contienen todos los datos y metadatos de una imagen, que siguen el formato mostrado en la Figura 2.5.

[1] `https://www.indexdatabase.de/db/s-single.php?id=96`

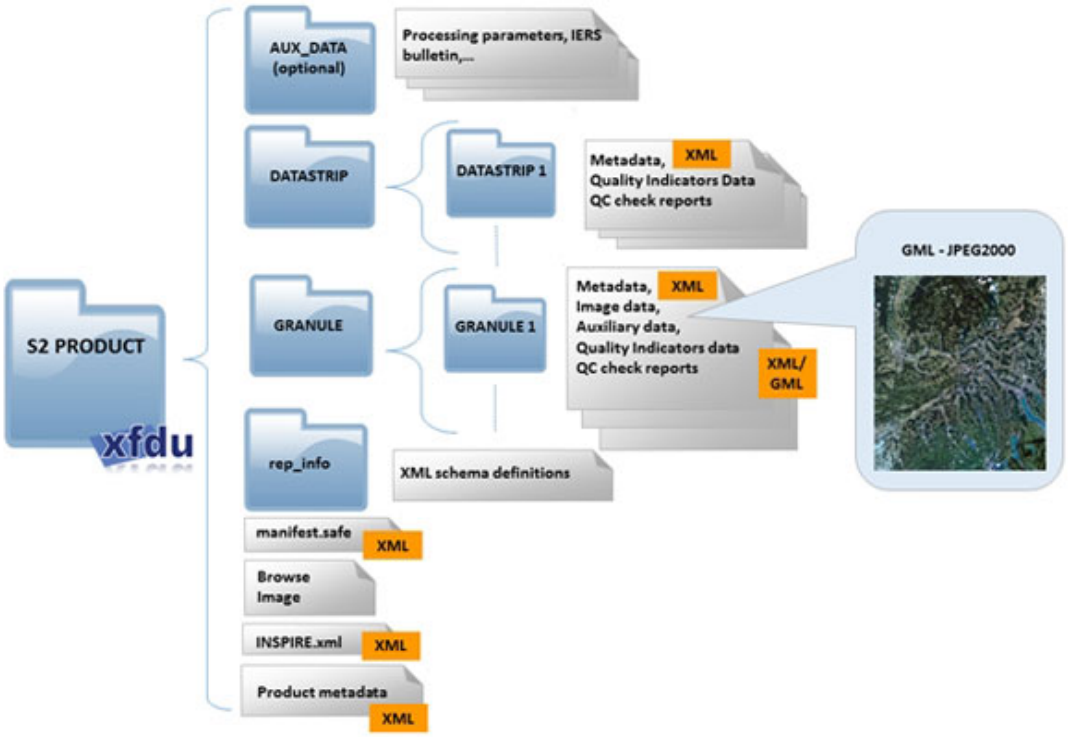


Figura 2.5: Datos y estructura que siguen los productos Sentinel-2

Fuente: `https://sentinel.esa.int/web/sentinel/user-guides/`

Algunos de los datos especialmente útiles que están contenidos en los productos son:

- **Bandas espectrales** (imagen multiespectral)
- Porcentaje de nubes y sombras de nube
- Porcentaje de nieve y hielo
- Fechas
- Coordenadas
- **Máscara de nubes**
- Metadatos de las bandas

2.1.3. Software

En este apartado se va a describir el software que ha facilitado el desarrollo del proyecto, sin tener en cuenta los más comunes (sistema operativo, consola de comandos, etc).

QGIS[2] es un Sistema de Información Geográfica de libre acceso, que tiene muchísimas utilidades para geografía, arqueología, etc. Esta herramienta permite trabajar con imágenes multiespectrales fácilmente. El uso que le damos en el proyecto es para preclasificar varias imágenes y así poder entrenar nuestro clasificador, lo que es posible gracias al plugin de clasificación semiautomática (Figura 2.6) para QGIS.

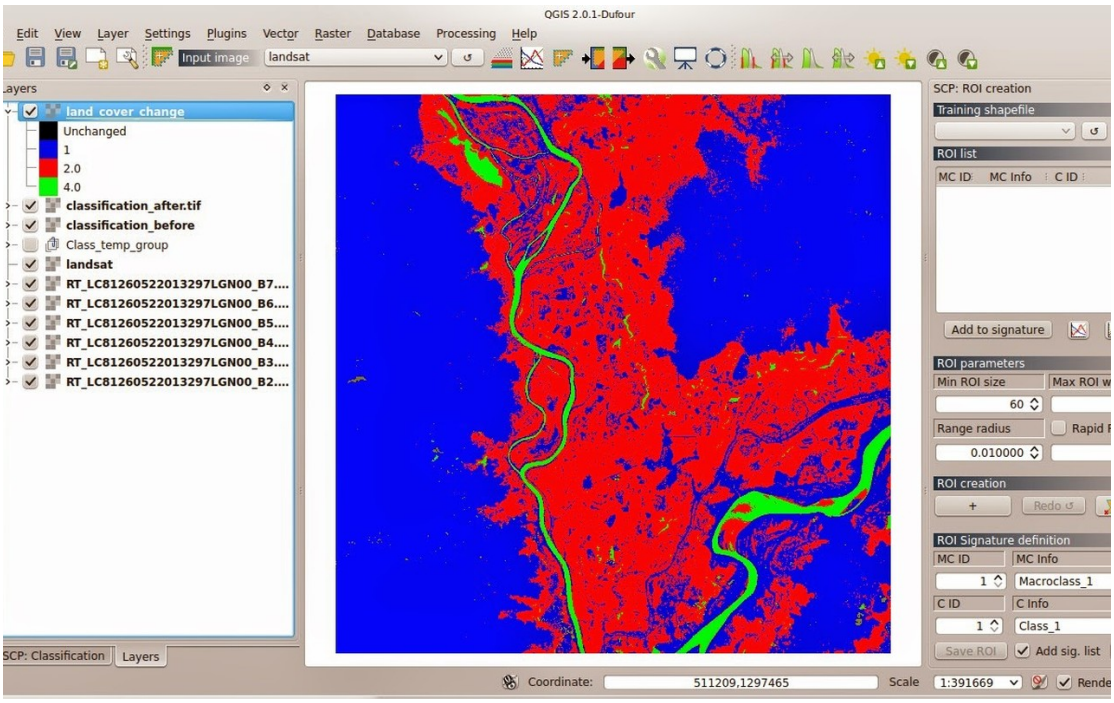


Figura 2.6: Ejemplo de preclasificación de una imagen multiespectral usando QGIS

Fuente: `https://fromgistors.blogspot.com/`

[2] `https://www.qgis.org/en/site/`

Copernicus open Access Hub es el servicio de acceso de datos principal de la misión Copernico (que contiene todas las misiones Sentinel, incluida Sentinel-2). Contiene una API para descargar los productos Sentinel (los datos y los metadatos de una imagen Sentinel) y una interfaz gráfica (Figura 2.7), que es más intuitiva para los menos familiarizados con la materia.

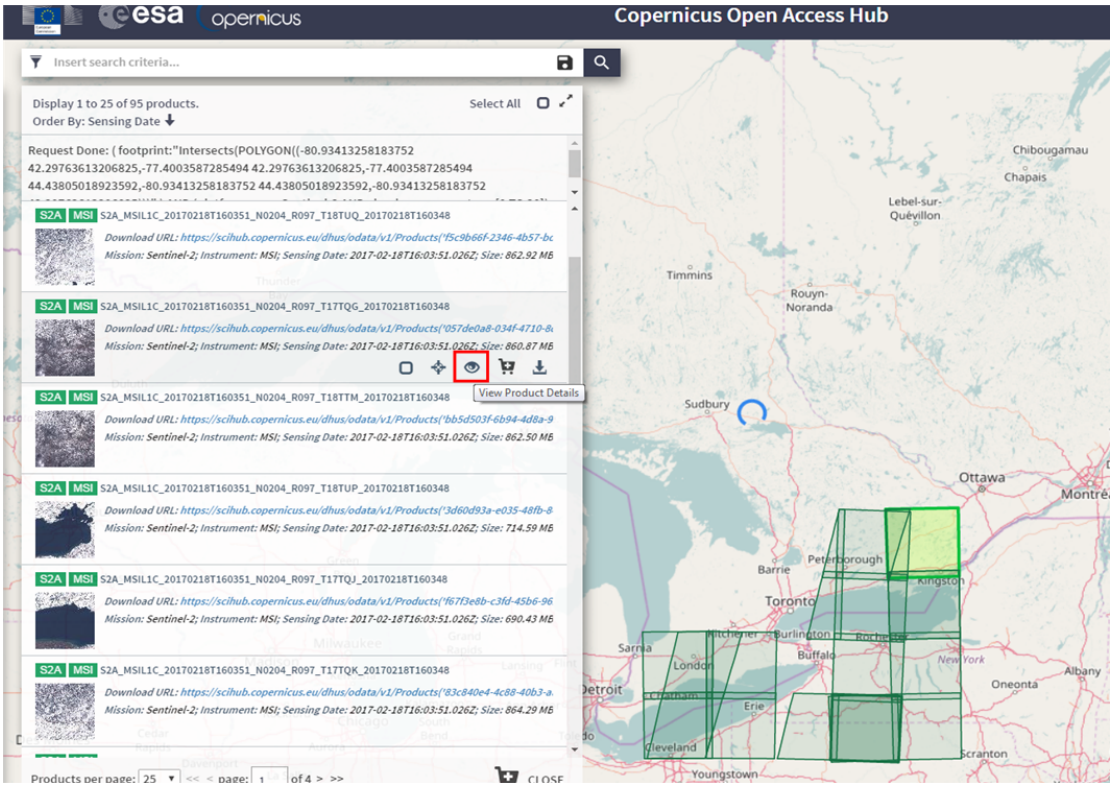


Figura 2.7: Interfaz gráfica para descargar los productos, el único requisito para ello es registrarse en la página web. Se pueden filtrar por coordenadas, fecha, satélite...

Fuente: `https://fromgistors.blogspot.com/`

2.2. Estado del arte

La clasificación de imágenes satélite sigue siendo en la actualidad una tarea bastante difícil, debido a su complejidad espacial y espectral. Esto se complica más debido a la heterogeneidad que se presenta en las áreas urbanas y paisajes suburbanos, que causan huecos entre las características identificables a bajo nivel.

En la última década, se ha puesto bastante esfuerzo en el desarrollo de métodos automáticos de clasificación para imágenes satélite. Éstos se dividen en clasificación basada en píxeles y en objetos [7], que clasifica agrupaciones de píxeles en vez de tratarlos por separado, como se puede ver en la Figura 2.8

Ambos métodos han tenido siempre limitaciones de precisión debido al ruido en la imagen y la alta varianza en cada clase. Para mejorar la precisión de los métodos basados en píxeles se suele realizar una postclasificación [2], aunque eso provoca la desaparición de pequeños detalles de pocos píxeles de tamaño en la imagen. Por las mismas razones, en los métodos basados en objetos el gran problema es la segmentación precisa de la imagen.

A día de hoy no hay ninguna solución efectiva para clasificación usando imágenes de satélites, por lo que se están intentando aplicar nuevas técnicas como *deep learning* [1] o redes convolucionales.

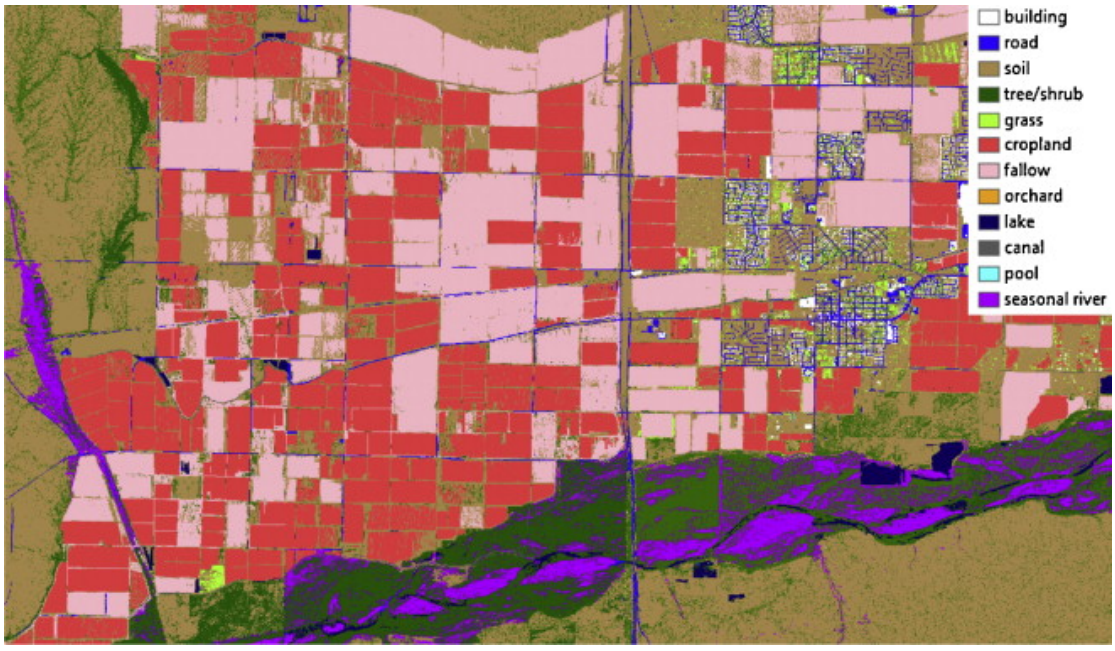


Figura 2.8: Clasificación de imágenes satélite basada en objetos

Fuente: Xiaoxiao Li et al. [4]

Capítulo 3

Metodología de desarrollo de software

La metodología a seguir para el desarrollo de este proyecto va a ser basada en iteraciones y inspirada en las metodologías ágiles [6]. Ya que es un área novedosa y requiere una gran cantidad de estudio, este modelo permite un alternado entre estudio de la materia y desarrollo del software al mismo tiempo (guiado por prototipos). Además, se irán sacando versiones funcionales cada cierto tiempo y se le irán añadiendo nuevas mejoras.

En cada iteración del proceso de desarrollo ágil se realizan las siguentes tareas (Figura 3.1):

1. Estudio de la materia
2. Reunión con los tutores para determinar las nuevas funcionalidades
3. Planificación
4. Desarrollar y probar

En las primeras reuniones se habló de la idea general del trabajo, definir objetivos y discutir el software que se usaría para el desarrollo del mismo. Se decidió empezar con QGIS, que nos permite obtener resultados visuales y es una buena primera toma de contacto con trabajos de teledetección, que es la adquisición de

información a través de sensores como los que usan los satélites Sentinel-2. Elegimos desarrollar el software en Python ya que al tener una gran comunidad activa y muchas librerías disponibles, resulta más fácil para el desarrollo de una aplicación en una materia en la que nunca antes se había trabajado, además de facilitar el prototipado y rápidas pruebas de concepto.

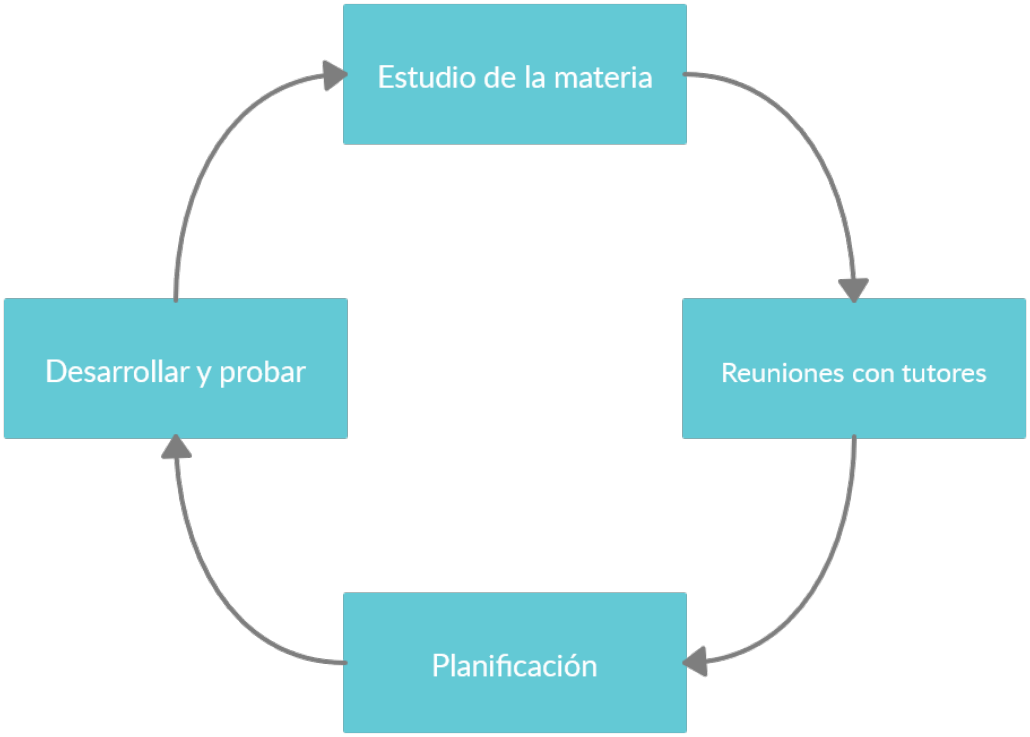


Figura 3.1: Metodología basada en iteraciones usada en el proyecto

Teniendo ya una primera versión básica y estable en Python, se habló en las siguientes reuniones de los problemas que aparecían respecto al tiempo de computación y del tamaño de los datos. Ya que en Python es difícil implementar un software paralelo usando *multithreading* debido al *Global Interpreter Lock*, se debatió el uso de la librería de Python **Dask**[1] que permite trocear los datos y los trata en distintos procesos. Esta idea se acabó descartando ya que se pierde el concepto de indexación y no se puede controlar el orden de los datos, que es necesario en nuestro caso.

[1] `https://dask.org/`

Finalmente se acabaron usando las librerías **multiprocessing** y **pandas** de forma conjunta para solventar ambos problemas, como se puede observar en la Figura 3.2. Tras realizar estos cambios, el tiempo de computación depende del número de núcleos del procesador y de la memoria RAM que se le permita a la herramienta. Además ya no habría problema con el tamaño de datos, lo que permite poder utilizar la herramienta para áreas de interés del tamaño que sea.

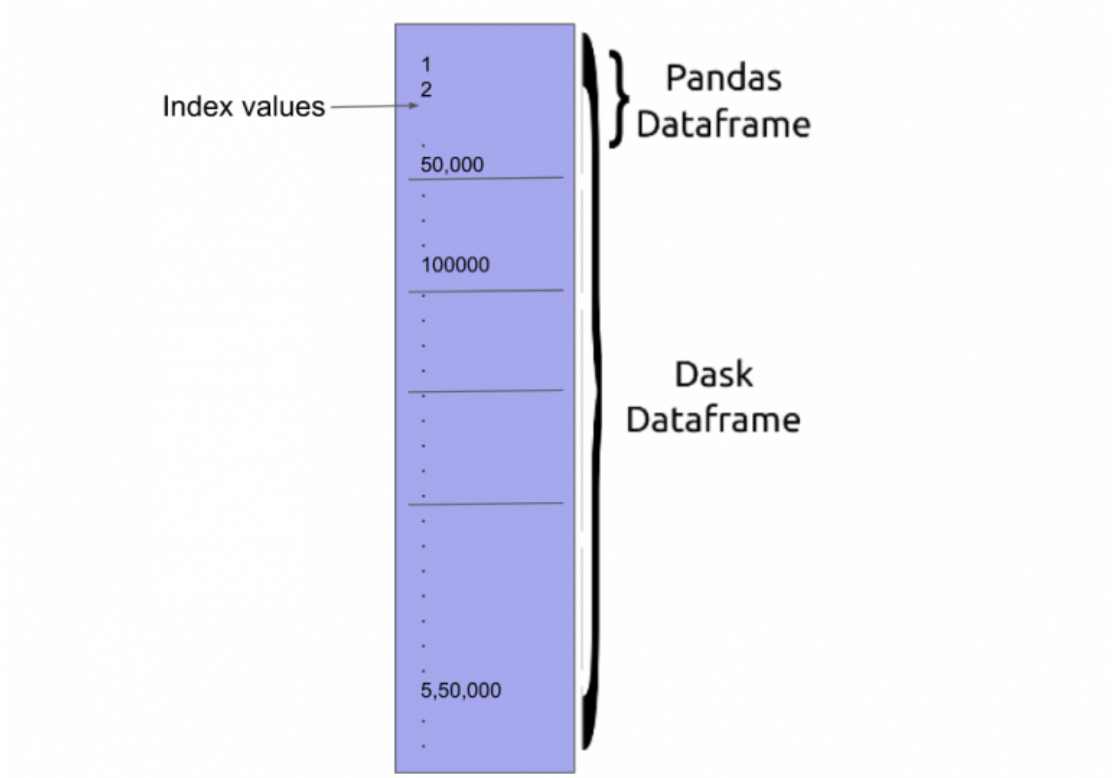


Figura 3.2: Comparación de una estructura Dask con una Pandas

Fuente: `https://www.analyticsvidhya.com/blog/`

Controlando a más bajo nivel la concurrencia con la librería *multiprocessing* podemos controlar el tamaño de las particiones, indexar sobre ellas, y mantener el concepto de orden en los datos.

Capítulo 4

Diseño e implementación de la herramienta

La herramienta que se ha desarrollado está dividida en varios módulos, siendo así fácilmente reutilizable y modificable. En este apartado vamos a comentar la arquitectura del software, estando cada módulo dividido en varios scripts.

El software implementado se organiza en 4 paquetes principales, cada uno atiende a una de las siguientes funciones:

1. Preparación del espacio de trabajo.
2. Descarga y preclasificación de las imágenes.
3. Preprocesamiento de datos.
4. Entrenamiento del clasificador y clasificación de datos.

Las fases 3 y 4 son completamente automáticas, las fases 1 y 2 requieren la descarga de las imágenes desde el servicio web del programa Copérnico y aportar varios datos de entrada, como las coordenadas del área a clasificar.

A continuación se va a explicar cada parte en detalle.

4.1. Preparación del espacio de trabajo

Lo primero de todo es crear una carpeta dentro del proyecto, donde van a estar todos los metadatos de la localización que vamos a clasificar. Como el proyecto tiene una estructura de rutas ya establecida, ejecutamos el siguiente código con el nombre de localización deseado.

```
location=example
mkdir -p $location/QGIS $location/geojson
cp pantano/paths.py $location
cp pantano/config.py $location
```

A continuación debemos crear un GEOJSON, que es un archivo JSON que define un área geográfica con nuestro escenario a analizar. Hay aplicaciones Web que nos permiten crearlo simplemente pinchando en un mapa y creando un polígono. Al obtenerlo lo añadimos en la carpeta localizacion/geojson/ previamente creada.

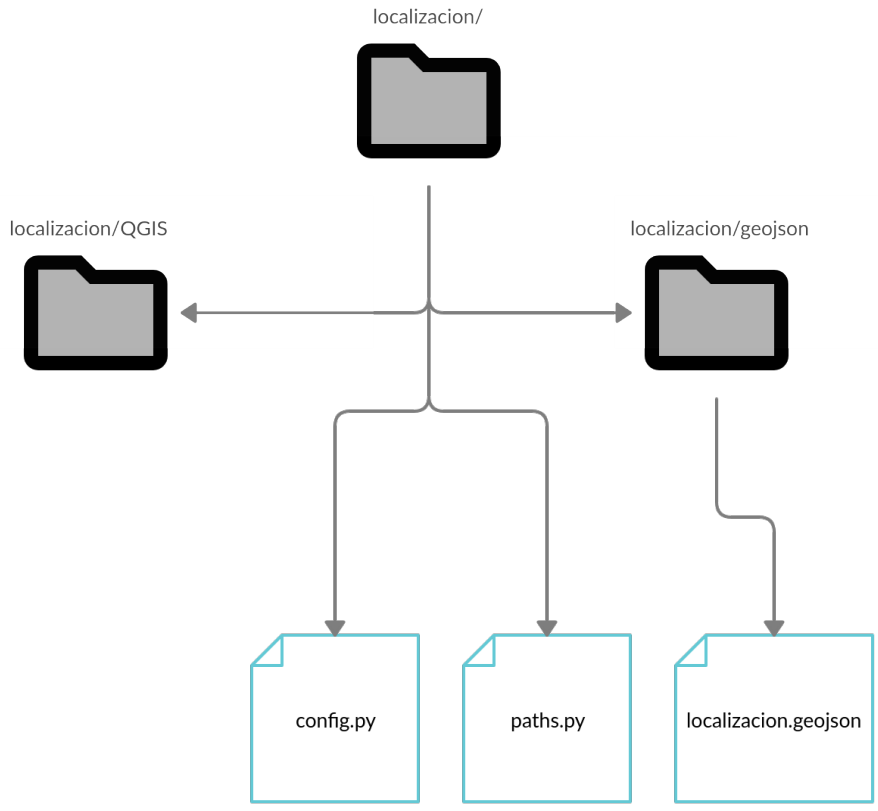


Figura 4.1: Estructura del espacio de trabajo para cada localización

Finalmente debemos modificar el archivo **paths.py** que se crea dentro de la carpeta. Este archivo contiene los directorios a los archivos que vamos a utilizar, por ejemplo el directorio donde se encuentran las imágenes que vamos a usar. La estructura final que habría quedado es la que se muestra en la Figura 4.1.

4.2. Descarga y preclasificación de las imágenes

Esta parte consiste en obtener las imágenes del área de interés usando la API que nos proporciona la ESA, normalizarlas para usarlas de entrada a nuestro programa y obtener una preclasificación de cada imagen (en nuestro caso usando QGIS). Esta preclasificación proporciona un etiquetado inicial de píxeles, y sobre ese etiquetado se apoyará la clasificación final.

4.2.1. Descarga de imágenes

La ESA proporciona una herramienta llamada **Copernicus Open Access Hub**, que fue mencionada en capítulos anteriores. Ya que queremos automatizar el proceso, vamos a usar la API en vez de la aplicación Web. Se nos proporciona además un script (**dhusget.sh**[1]) que facilita el uso de la API, siendo necesario solo añadir las coordenadas de la región de interés, el usuario y contraseña (previamente es necesario registrarse) y el número de productos que queremos. El formato final de uso sería el siguiente:

Código fuente 4.1: Uso de dhusget.sh

```
./dhusget.sh -T S2MSI2A -o product -u <user> -p <password>
   -c <coordinates> -l <n_products>
```

Este script descarga el número de productos que necesitemos; el problema es que vienen comprimidos y contienen muchos archivos que no vamos a necesitar y ocupan bastante espacio. Para solventarlo tenemos un script que organiza todos los productos y toma solo lo necesario de cada imagen, **prepare_products.sh**

[1] `https://scihub.copernicus.eu/userguide/BatchScripting`

A continuación usamos **prepare_products.sh**, que nos descomprime y normaliza las imágenes al formato que usa la herramienta como entrada. Para cada imagen se extraen únicamente las bandas que usamos, y se agrupan en una carpeta nombrada con su fecha.

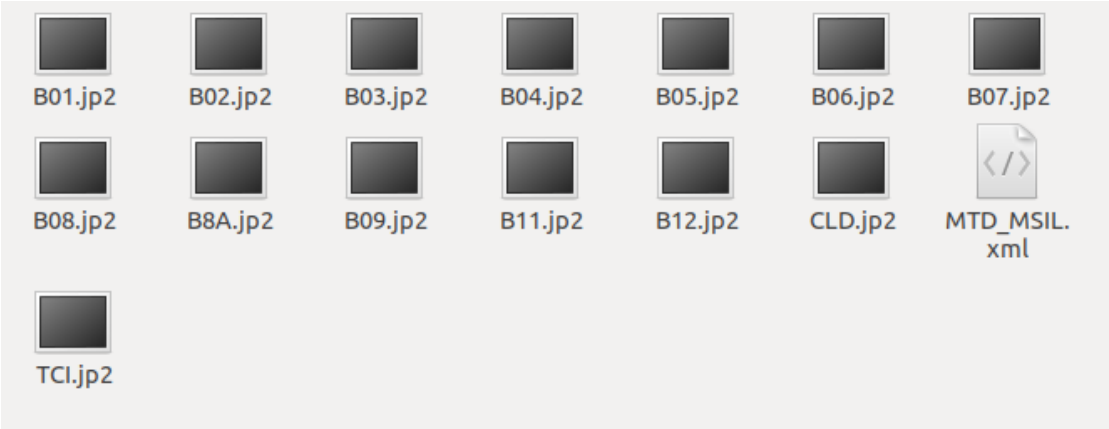


Figura 4.2: Ejemplo de imagen normalizada tras prepare_products.sh

Donde **B01 - B12** son las 12 bandas de la imagen, **CLD** es la máscara de nubes, **TCI** es la imagen RGB (para visualización) y **MTD_MSIL** son os metadatos de la imagen.

4.2.2. Preclasificación de las imágenes

Ya que la herramienta usa técnicas de aprendizaje supervisado, es necesario *describirle* las clases que vamos a querer clasificar. Para facilitar el uso del software, tenemos **generate_script_QGIS.py**. Un script en Python que genera un código con instrucciones secuenciales que se puede usar en QGIS. Este código nos genera una imagen de preclasificación para cada imagen del área de interés que se han descargado en el paso anterior.

Para describir las clases que queremos clasificar debemos abrir una imagen en QGIS de las que previamente hemos descargado y marcar un área que corresponda a esa clase. Cada clase se va describir con la firma espectral del área que hemos marcado. Cuando hayamos marcado todas las clases que vamos a querer diferenciar en la imagen, QGIS crea un archivo **.scp**.

Al tener ese archivo ejecutamos en QGIS el código que hemos comentado previamente, lo cual nos genera una imagen preclasificada (con errores que vamos a corregir en la clasificación final) para cada imagen descargada. Estas imágenes se generan asignándole a cada píxel la firma espectral (de las áreas anteriores) que más se parece a la suya.

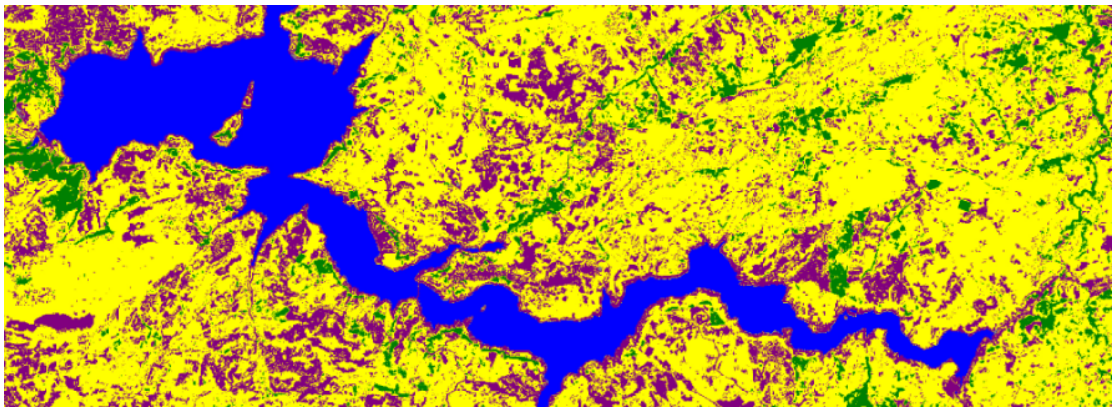


Figura 4.3: Imagen preclasificada del embalse de Íznajar, Córdoba

Como resultado, obtenemos los datos etiquetados, es decir, los atributos correspondientes al conjunto de datos y sus etiquetas.

4.3. Preprocesamiento de datos

La herramienta trata de entrenar un clasificador con las imágenes que hemos obtenido previamente. Esto implica un previo preprocesamiento de las imágenes, ya que las necesitamos en formato **.csv** o similar. Además nos encontramos con el problema de las nubes que tienen propiedades heterogéneas y son difíciles de clasificar, por lo que la mejor opción es no tenerlas en cuenta y eliminarlas de los datos de entrenamiento.

Como las clases que usamos en los datos vienen de las imágenes preclasificadas anteriormente, que tienen un notable porcentaje de errores. Por tanto necesitamos usar técnicas de detección de *outliers*. Todo esto se va a comentar fase por fase a continuación, en el mismo orden que lo hace la herramienta.

4.3.1. De imagen multiespectral a tabla de datos

Ya que vamos a usar técnicas de aprendizaje supervisado, necesitaríamos una tabla con varias columnas de datos y una con la clase de cada fila de datos. Para ello necesitamos transformar nuestras imágenes a una tabla, ya que las bandas de las imágenes están en formato **.jp2** (un formato de compresión para imágenes con gran resolución); para esta tarea vamos a usar la librería **rasterio**[2] para Python. Esta librería nos permite leer las imágenes en ese formato como arrays de tipo numpy (extensión de Python que facilita el tratamiento de matrices), y además recortarlas con la forma de un GEOJSON. Ya que las bandas no tienen la misma resolución espacial, hay que realizar un reescalado a una resolución en común. Dependiendo del tamaño de la imagen y de la precisión que necesitemos, podemos usar una resolución espacial mayor o menor.

Teniendo ya los arrays vamos a crear dos conjuntos de datos en formato *dataframe*:

- Uno usando valores de reflectancia (firma espectral) que tendría 16 columnas (fecha de la imagen, latitud y longitud del píxel, 12 bandas espectrales y la clase asignada).
- Otro usando índices de vegetación, que son más representativos, y reducen el número de columnas combinando varias bandas (menos datos que procesar). En este caso tendríamos 10 columnas (fecha de la imagen, latitud y longitud del píxel, 6 índices de vegetación y la clase asignada).

[2] `https://rasterio.readthedocs.io/en/latest/`

Como resultado, esta tabla de datos sigue un formato que es fácilmente convertible a formatos típicos como CSV, JSON, TSV, XML, ect.

Se crean dos conjuntos porque si el conjunto de datos con los índices de vegetación tiene buenos resultados, podríamos descartar el otro para una versión final. Esto podría ser interesante ya que reduce el tamaño de los datos que tienen que ser tratados y por tanto, el tiempo de computación. Los índices que usemos en la tabla también influyen en el resultado, ya que tendremos que elegir unos que distingan bien las clases que queramos distinguir. Todo esto se comentará en los siguientes capítulos.

4.3.2. Detección de outliers

Los **outliers** son valores atípicos que aparecen en el conjunto de datos y no concuerdan con la clase a la que pertenecen respecto al resto de datos de esa clase. Ya que la preclasificación que realizamos no está libre de errores y no queremos entrenar mal al clasificador, es necesario marcar los datos que tienen alta probabilidad de no tener la clase correcta asignada. Puesto que el estudio de un conjunto de datos con tantas dimensiones sería bastante costoso, se usa la **distancia de Mahalanobis** (Ecuación 4.1) para ver los puntos que menos representan cada clase [3]. La distancia de Mahalanobis es como la distancia euclídea pero cada dimensión está ponderada con diferentes valores. La idea es calcular los centroides de cada clase y la distancia de Mahalanobis de cada punto al centroide de su clase. Entonces marcamos como outlier un porcentaje de los datos (con mayor distancia) de cada clase.

$$DM(x) = \sqrt{(x - \bar{x})^T \, \Sigma^{-1} \, (x - \bar{x})} \tag{4.1}$$

donde:

x = Vector de datos de entrada.

$\bar{x}$ = Centroide del conjunto de datos.

Σ = Matrix de covarianza (covarianzas entre todas columnas de los datos).

Podemos tener una visualización de los píxeles marcados ejecutando **outliers_plot.py**, que nos muestra los píxeles marcados como outliers en las imágenes.

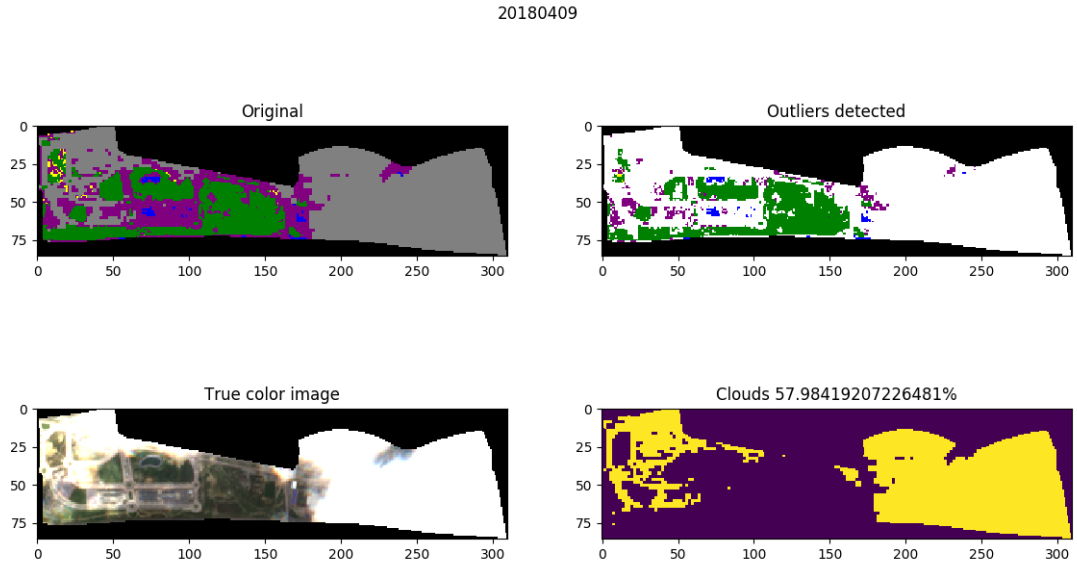


Figura 4.4: Resultado al ejecutar outliers_plot.py sobre una imagen, el área mostrada es Teatinos, Málaga

En la Figura 4.4 podemos ver que se están marcando como outliers los píxeles que tienen ruido causado por la sombra de la nube y están probablemente mal preclasificados. Cabe mencionar que los píxeles que están cubiertos por la máscara de nube son directamente marcados como outliers. Finalmente nos quedamos con los píxeles que probablemente están bien clasificados, y que hemos rescatado de una imagen que a priori tiene poca información.

Esta herramienta nos sirve para visualizar el comportamiento del algoritmo que detecta los outliers, en nuestro caso la distancia de Mahalanobis. Además podemos usarla cuando cambiemos algún parámetro referido a la detección de outliers, que están contenidos en el archivo **config.py** de cada localización.

4.4. Entrenamiento del clasificador y clasificación de datos

Aunque el modelo de clasificación se puede modificar en el archivo **config.py** (como otros muchos parámetros), el modelo por defecto que se ha implantado es un metaclasificador. Es decir, un clasificador que combina la salida de otros clasificadores.

Entrenamos independientemente 3 modelos típicos de aprendizaje supervisado: un clasificador *Naive Bayes*, un *KNN* y un *Random Forest*. Posteriormente, cuando vayamos a clasificar un nuevo dato, los tres clasificadores obtendrán una salida. Por tanto, la clase final elegida será la que tenga más votos entre los tres modelos básicos. Este metaclasificador se puede configurar para decidir los triples empates o para ponderar la importancia de los clasificadores que usa, entre otras muchas opciones.

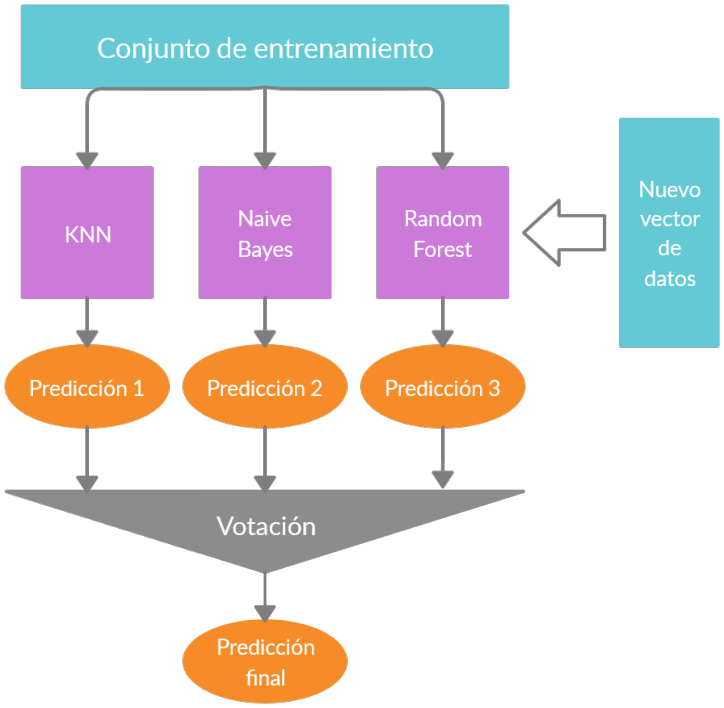


Figura 4.5: Diagrama clasificador usado por defecto

En la parte de clasificación hemos usado dos librerías de Python: **Mlxtend**[3] para el metaclasificador y **Sklearn**[4] para los modelos de aprendizaje supervisado.

Como resumen final, se muestra la lista de scripts implementados y su función:

Nombre	**Lenguaje**	**Función**
prepare_products	Bash	Normaliza los productos descargados
ggplot	R	Visualización de datos (Figura 5.9)
generate_script_QGIS	Python	Automatiza la preclasificación en QGIS
generate_dataset_multiprocessing	Python	Transforma las imágenes a dataframe
outliers_plot	Python	Visualización de outliers (Figura 4.4)
detect_outliers_dataset	Python	Marca los outliers en el dataframe
utils	Python	Funciones que son usadas por varios scripts
classification	Python	Entrena el clasificador
prediction	Python	Clasifica nuevas imágenes
paths	Python	Contiene constantes con directorios
config	Python	Contiene variables de configuración

[3] `http://rasbt.github.io/mlxtend/`
[4] `https://scikit-learn.org/stable/`

Capítulo 5

Resultados

Para mostrar el funcionamiento de la herramienta y algunos casos de uso, se han realizado dos experimentos en dos áreas de diferentes características en Málaga.

- **Teatinos**: el barrio de Teatinos es una zona urbana a las afueras de Málaga. Aunque un gran porcentaje del área son zonas metropolitanas, hay bastantes explanadas y zonas verdes.
- **Pantano de Iznájar**: En la frontera entre Córdoba y Málaga se encuentra el pantano de Iznájar, un embalse de gran magnitud construido en los años 60. Se encuentra en una zona rural, hay pequeños pueblos a su alrededor pero la mayor zona está cubierta por tierra de cultivo.

Ambos se han realizado usando el mismo procedimiento (explicado en el capítulo previo), con los datos obtenidos se han realizado algunas gráficas para visualizar y entender mejor los resultados. Además se van a comentar varios problemas que nos encontramos al desarrollar esta herramienta y la solución que se ha usado.

5.1. Problemas y soluciones

Antes de mostrar los resultados obtenidos, se van a comentar varios problemas que han surgido al desarrollar la herramienta.

5.1.1. Nubes

Uno de los grandes problemas en teledetección son las nubes o sus sombras. Cada vez que una nube se cruza entre el sensor y el área a estudiar se produce ruido en la imagen, por tanto los píxeles afectados ya no son clasificables. Esto nos obliga a calcular una máscara de nube que nos indique si un píxel está afectado por ese ruido o no. Esto es bastante complejo por la infinidad de formas en las que se pueden presentar las nubes (depende de la densidad, la altura, etc), como se puede observar en la Figura 5.1. Además, hay que tener en cuenta las sombras, ya que eso significa que una nube se encuentra entre el emisor de luz (Sol) y el área a estudiar.

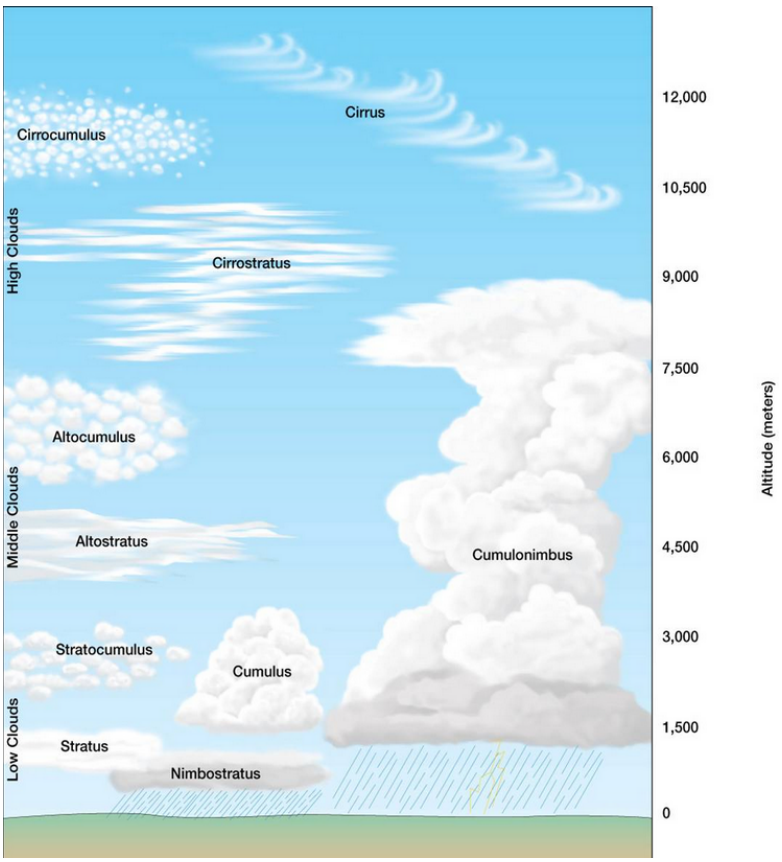


Figura 5.1: Diferentes tipos de nubes

Fuente: `https://www.almanac.com/content/types-clouds`

Durante el desarrollo de la herramienta se han probado varios algoritmos para enmascarar las nubes, y los resultados eran bastante malos ya que algunas técni-

cas mostraban muchos falsos positivos que nos eliminan píxeles sin ruido, otras técnicas producen demasiados falsos negativos, mientras que otras funcionaban bien pero solo con las imágenes TOA (como es el caso de la librería sen2cor de Python). Finalmente decidí usar la máscara de nubes que se proporciona al descargar las imágenes (CLD.jp2), que tiene bastantes falsos negativos pero al ver los resultados, dichos píxeles se corrigen al detectar los outliers.

5.1.2. Índices de espectrales

Como se ha comentado previamente, vamos a tratar una gran cantidad de datos, e intentamos reducirlos comprimiendo las bandas espectrales en índices espectrales. El problema es decidir cuáles vamos a usar, y cuántos son necesarios para una buena clasificación. Obviamente menos de 12, ya que si no estaríamos aumentando los datos y dando información redundante. Tras varias pruebas se puede ver que depende de las clases que hayamos definido. Por ejemplo, si tenemos la clase vegetación lo normal es usar el NDVI que la representa bien, para la clase agua usamos el NDWI[1], etc. Mientras más clases haya que diferenciar más índices añadimos, pero si queremos diferenciar un par de clases para segmentar el área, con dos o tres es suficiente.

5.2. Área de zonas verdes en Teatinos, Málaga

Este estudio pretende demostrar las capacidades de la herramienta y su funcionamiento. Vamos a aplicar la herramienta sobre Teatinos (actualmente zona universitaria de Málaga) en un intervalo de tiempo de aproximadamente tres años. Distinguiremos entre vegetación (área verde), construcciones, agua y tierra seca (tierra sin vegetación).

Al finalizar la ejecución del programa obtenemos dos tipos de salidas. La primera es un conjunto de datos (Figura 5.2) con las coordenadas de cada píxel clasificado

[1] `https://www.sentinel-hub.com/eoproducts/ndwi-normalized-difference-water-index`

(durante el intervalo de tiempo usado). La otra salida son esos mismos datos pero en imagen, para visualizar la clasificación.

	DATE	LONGITUDE	LATITUDE	CLASS
1	20180330	36.719930306000002	-4.501395201200000	water
2	20180330	36.719846251200003	-4.501395201200000	water
3	20180330	36.719762196399998	-4.501395201200000	water
4	20180330	36.719678141599999	-4.501395201200000	water
5	20180330	36.719594086800001	-4.501395201200000	water
6	20180330	36.719510032099997	-4.501395201200000	water
7	20180330	36.717240553000003	-4.501395201200000	water
8	20180330	36.717156498200005	-4.501395201200000	built
9	20180330	36.717072443500001	-4.501395201200000	built

Figura 5.2: Salida en formato csv

En el archivo **config.py** se pueden modificar los colores de cada clase y si quieres mostrar o no los píxeles detectados como nube.

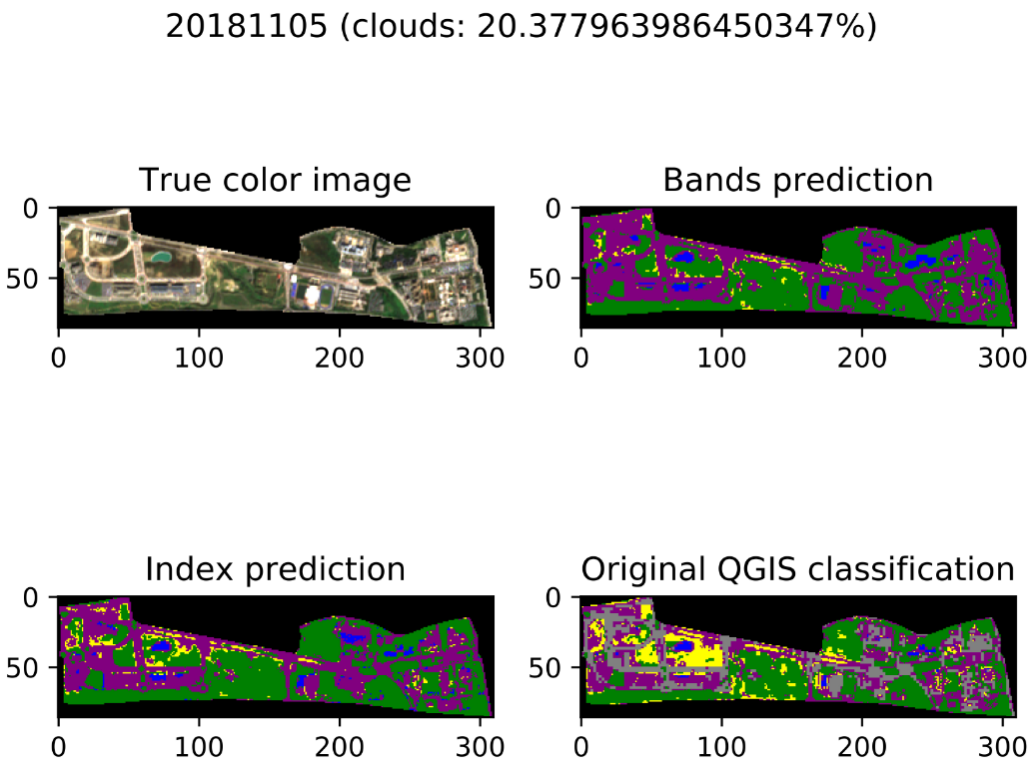


Figura 5.3: Salida en formato jpg
(verde: vegetación, amarillo: tierra, azul: agua, morado: contrucción)

La Figura 5.3 muestra la fecha correspondiente a la imagen, la imagen RGB, la predicción usando el conjunto de datos de bandas, la predicción índices espectrales y la preclasificación original. Mostramos este cuadro ya que para el desarrollo de la herramienta ha sido útil, al tener facilidad para comparar resultados. Cabe destacar que es fácilmente modificable y esta no tiene por qué ser la salida final para el usuario.

Como podemos observar, en el ejemplo anterior hemos decidido no mostrar los píxeles marcados como nube (en la imagen preclasificada los píxeles marcados en gris son detectados como nube). Al ser éstos falsos positivos, se clasifican correctamente en la imagen final. Esto no sale tan bien cuando usamos una imagen que sí tiene nubes.

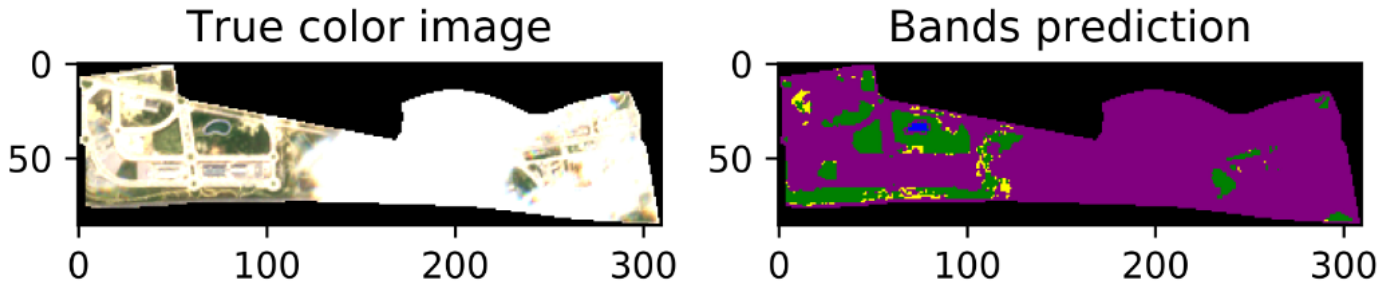


Figura 5.4: Imagen de Teatinos en la que aparecen nubes

En estos casos debemos mostrar la máscara de nubes, ya que no nos importan los falsos positivos. Se puede ver la importancia de una máscara de nubes sin errores, ya que toda la clasificación depende de ella.

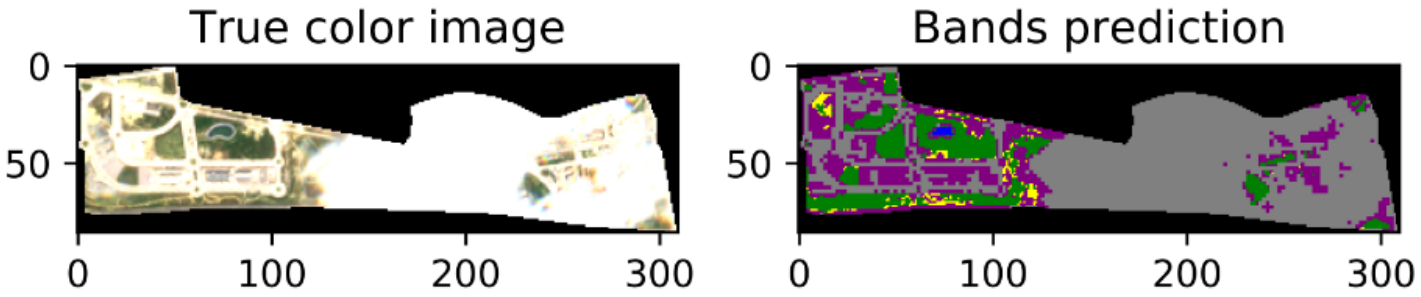
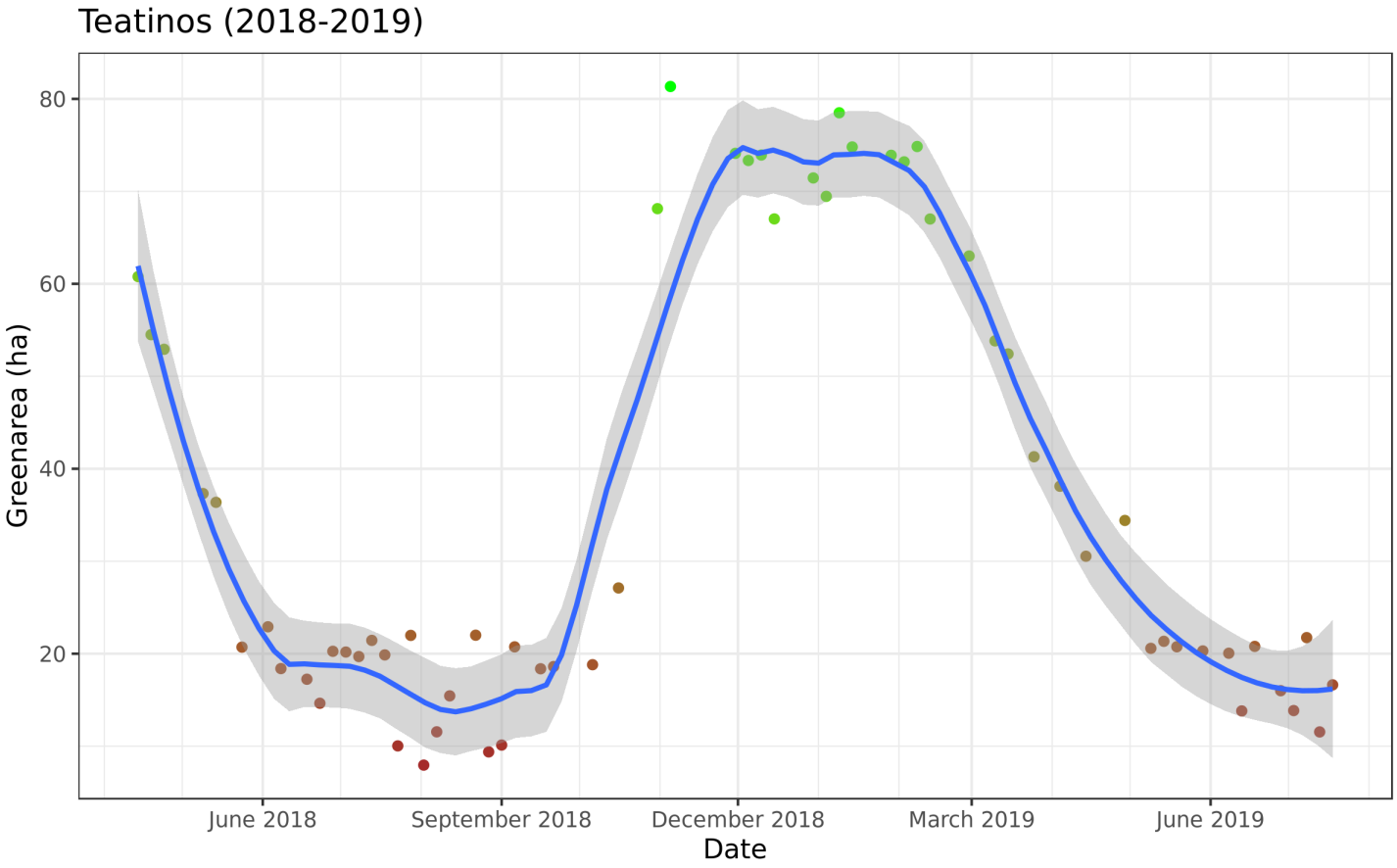


Figura 5.5: Clasificación incluyendo la máscara de nubes

Teniendo ya todas las imágenes clasificadas, podemos obtener fácilmente el porcentaje de zonas verdes en Teatinos contando los píxeles de vegetación que hay en cada imagen. Cada píxel son $100m^2$, ya que la resolución espacial de las bandas es 10m. Si visualizamos el área de zonas verdes en Teatinos en el último año, podemos ver que sigue un patrón:



Podemos observar los datos corresponden con la realidad (mínimos en verano y máximos en invierno). Si realmente detectamos un patrón podríamos intentar predecir mediante series temporales las zonas verdes en el futuro. Por tanto vamos a ver si en los tres últimos años se ha seguido el mismo patrón. El único problema para realizar esto es la gran cantidad de datos que tenemos que descargar, ya que ocupan mucho espacio en memoria.

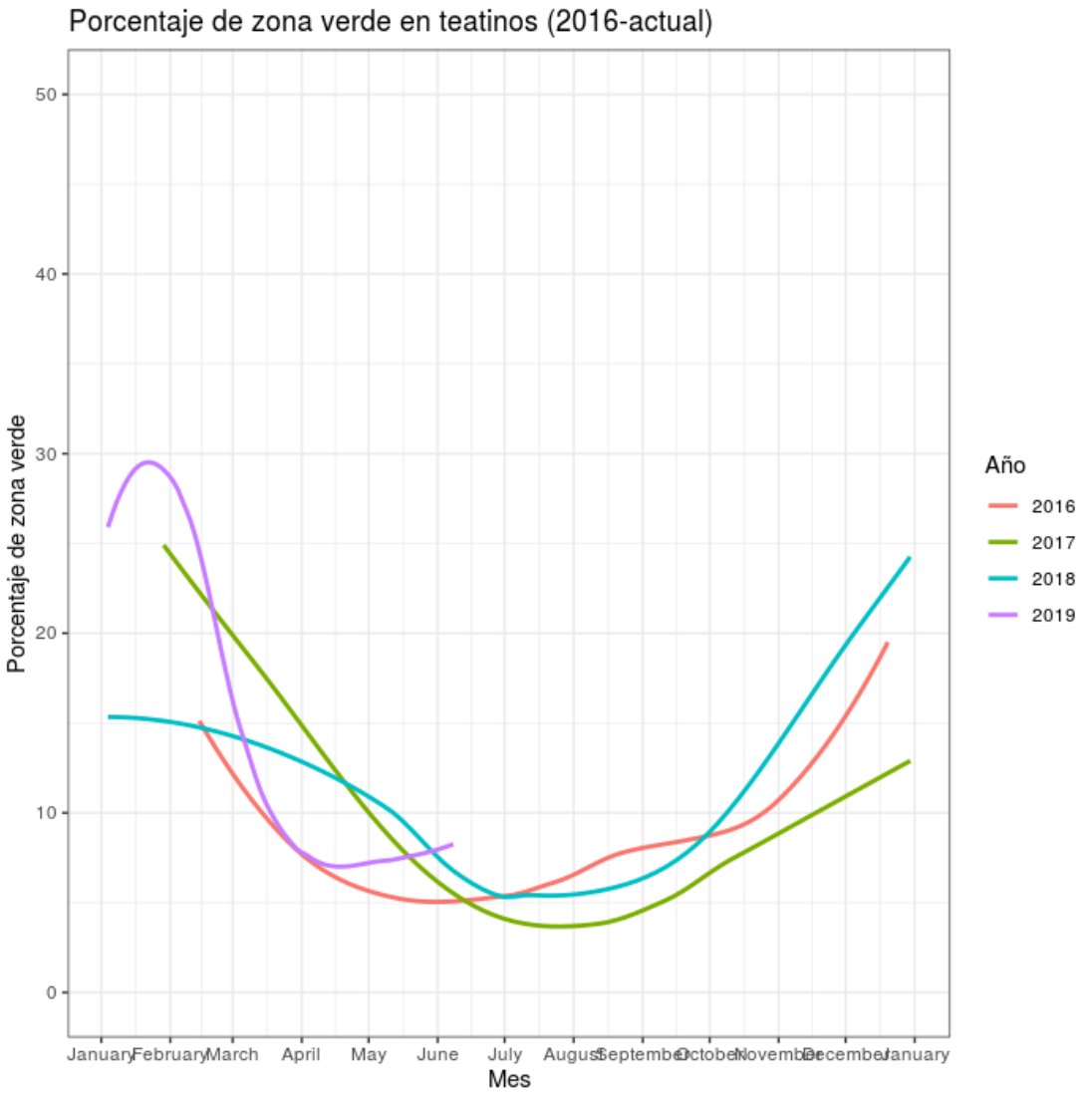


Figura 5.6: Porcentaje de zonas verdes en el área de Teatinos en los últimos 3 años

Se ve claramente que todos los años se ha seguido el mismo patrón antes comentado. Hay que tener en cuenta que las imágenes con más de un 10 % de nubes no se han usado.

Aunque no vamos a realizar predicciones usando las series temporales ya que queda fuera del propósito de esta memoria, se muestra el tipo de estudios que se pueden realizar usando esta herramienta y lo útil que puede llegar a ser.

5.3. Superficie del embalse de Iznájar, Córdoba

Para mostrar como reacciona la herramienta ante superficies más grandes, vamos a medir la superficie del embalse de Iznajar (ya que no sabemos la profundidad no podemos calcular el volumen). El procedimiento es el mismo que para Teatinos, vamos a clasificar las imágenes que hemos descargado (en este caso desde junio de 2018 hasta septiembre de 2019), y sumar los píxeles que se clasifiquen como agua (cada píxel son 100m^2 en el caso de Sentinel-2).

Cabe destacar que estas imágenes son bastante grandes (25km^2, imágenes de $1000x2500$ píxeles) y el tiempo de clasificación puede ser de hasta media hora por imagen, dependiendo del hardware del que dispongamos. Esto se solucionaría incluyendo una opción para reescalar las imágenes, que nos permitiera reducir el tamaño de los datos, así como el tiempo de computación. Aunque esto causa una pérdida de exactitud en los resultados, con superficies tan grandes el cambio no sería significativo.

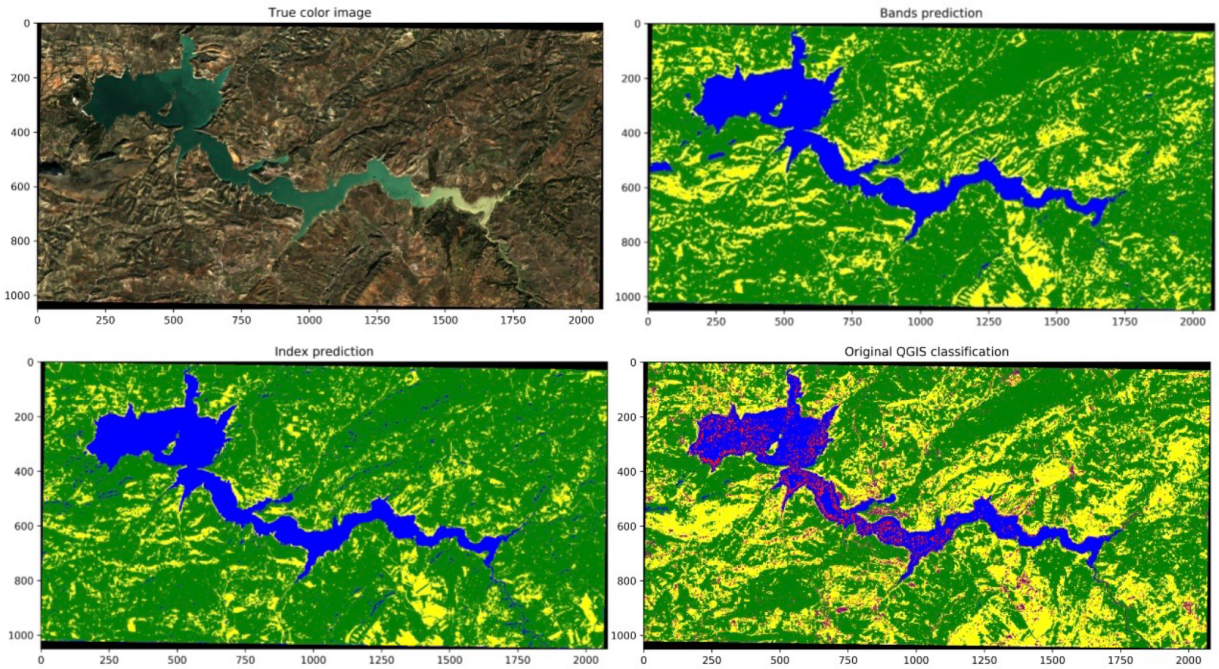


Figura 5.7: Ejemplo de imagen del área del embalse de Iznájar clasificada

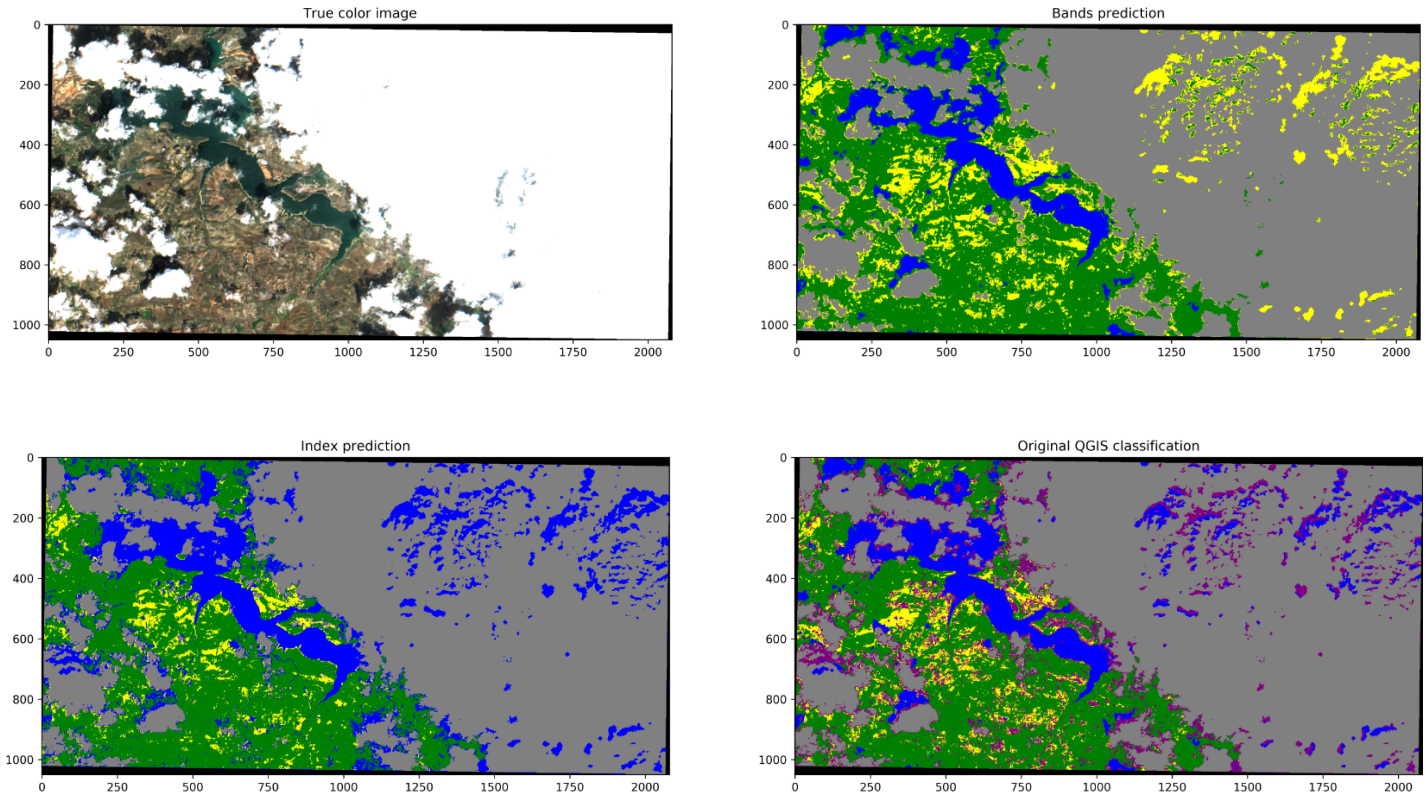


Figura 5.8: Ejemplo de imagen desechada por la aparición de nubes

Podemos ver de nuevo la importancia de la máscara de nubes, ya que los falsos negativos causan errores en la clasificación. Esto provoca que tengamos que desechar la imagen al completo, aunque haya partes que han sido clasificadas correctamente.

Volviendo a los resultados obtenidos, como en el caso anterior obtenemos un archivo .csv junto con las imágenes clasificadas (Figuras 5.7 y 5.8). Usando esos datos, podemos realizar una gráfica con el área de agua que hay en las imágenes (que en este caso coincide con la superficie de agua embalsada).

Para verificar los datos, se ha buscado el volumen de agua que realmente hay en el embalse. Si nos fijamos en el mismo intervalo del tiempo, podemos observar que el área y el volumen son proporcionales:

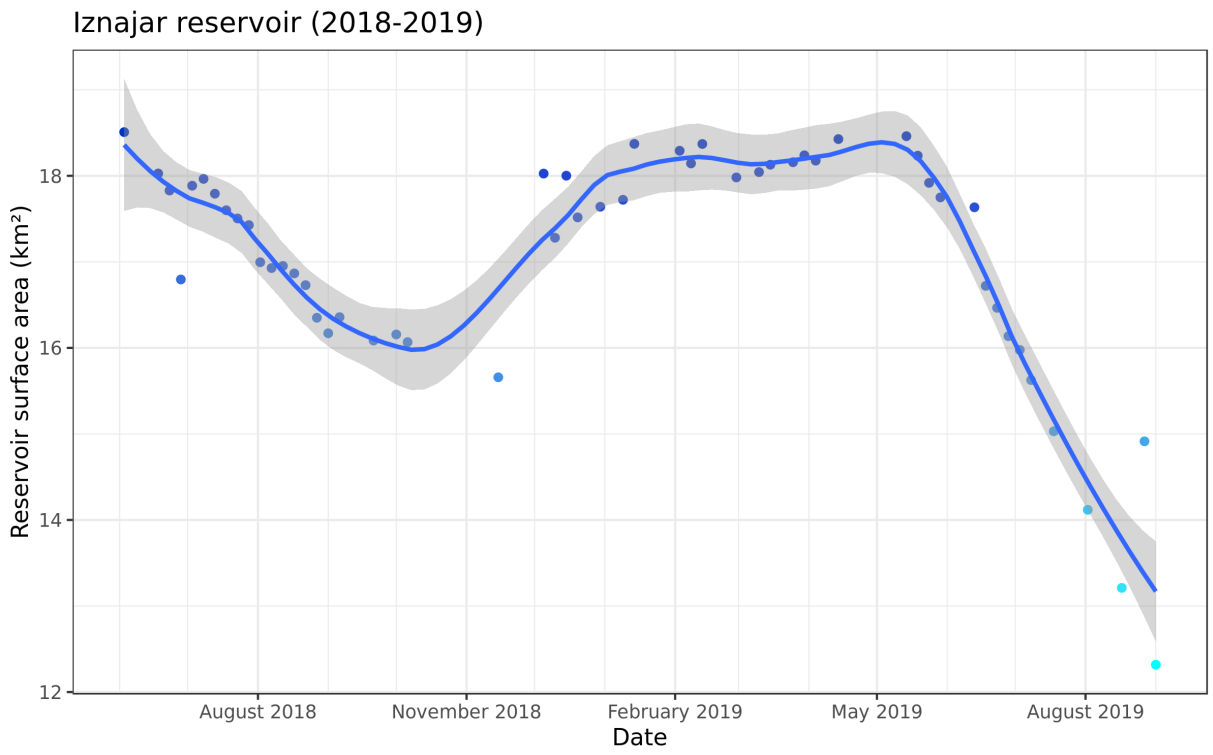


Figura 5.9: Área del embalse de Iznájar (junio 2018 - septiembre 2019)

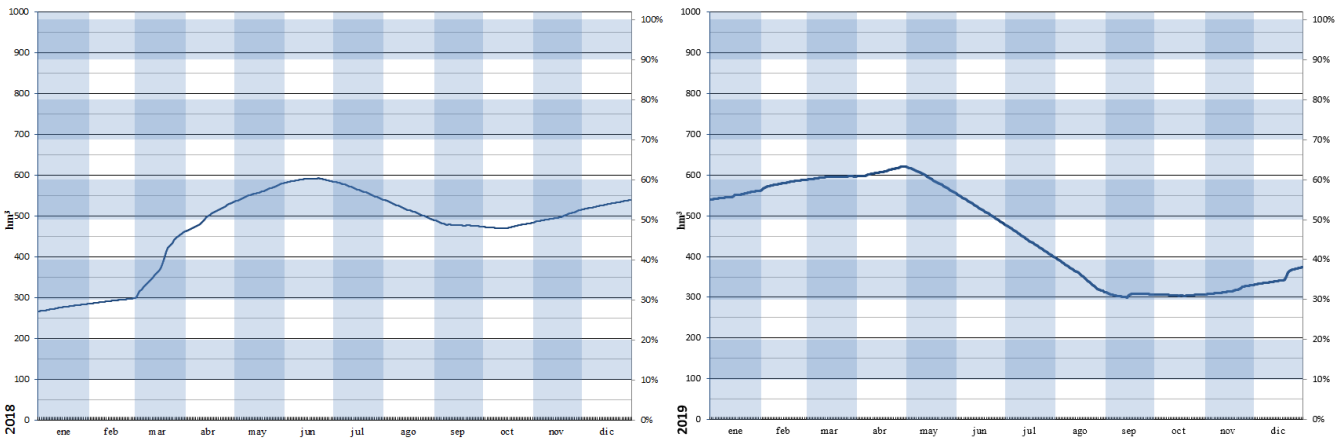


Figura 5.10: Volumen del embalse de Iznájar (2018 y 2019)

Podemos imaginar la cantidad de casos de uso que le podemos dar a la herramienta y el interés que puede tener para realizar estudios similares.

Capítulo 6

Conclusiones y trabajo futuro

Se puede concluir que para estudiar la superficie terrestre usando imágenes satélite correctamente falta aún mucho trabajo de investigación, desarrollar mejores algoritmos, desplegar mejores satélites (en 2020 se pondrán en órbita los satélites Sentinel-6), etc.

Es un campo que aún tiene mucho por delante pero que tiene mucho potencial por explotar. En la actualidad se están desarrollando predictores de incendios y otras catástrofes naturales usando este método. Cada vez los satélites tienen bandas más potentes (térmicas, dedicadas exclusivamente para detectar nubes, ...), y se va aumentando la resolución espacial a medida que las capacidades de cómputo crecen.

A nivel personal, este proyecto puede mejorar en varios ámbitos, implementarlo en un servidor que descargue, clasifique y procese automáticamente las imágenes a medida que se van tomando, mejorar la máscara de nubes, etc. Finalmente cabe destacar que la herramienta y una guía de uso detallada se encuentran en un repositorio de GitHub[1] abierto para cualquiera que tenga interés en ella.

[1] `https://github.com/AmbroxMr/TFG.git`

Bibliografía

[1] Joint deep learning for land cover and land use classification. *Remote Sensing of Environment*, 221:173 – 187, 2019.

[2] Cakir H.I. Nelson S. a C. Khorram S. Hester, D.B. Per-pixel classification of high spatial resolution satellite imagery for urban land-cover mapping. *Photogramm. Eng. Remote. Sens.*, 2008.

[3] Christophe Leys, Olivier Klein, Yves Dominicy, and Christophe Ley. Detecting multivariate outliers: Use a robust variant of the mahalanobis distance. *Journal of Experimental Social Psychology*, 74:150 – 156, 2018.

[4] Xiaoxiao Li, Soe W. Myint, Yujia Zhang, Chritopher Galletti, Xiaoxiang Zhang, and Billie L. Turner. Object-based land-cover classification for metropolitan phoenix, arizona, using aerial photography. *International Journal of Applied Earth Observation and Geoinformation*, 33:321 – 330, 2014.

[5] Julien Radoux, Guillaume Chomé, Damien Jacques, Francois Waldner, Nicolas Bellemans, Nicolas Matton, Céline Lamarche, Raphael d'Andrimont, and Pierre Defourny. Sentinel-2's potential for sub-pixel landscape feature detection. *Remote Sensing*, 8:488, 06 2016.

[6] Ken Schwaber. *Agile Project Management With Scrum*. Microsoft Press, USA, 2004.

[7] A. Suliman and Y. Zhang. Optical-elevation data co-registration and classification-based height normalization for building detection in stereo vhr images. *Advances in Remote Sensing*, 6, 2017.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga